**Noname manuscript No.**
(will be inserted by the editor)

# Constraint-based Clustering Selection

**Toon Van Craenendonck · Hendrik Blockeel**

Received: date / Accepted: date

**Abstract** Clustering requires the user to define a distance metric, select a clustering algorithm, and set the hyperparameters of that algorithm. Getting these right, so that a clustering is obtained that meets the users subjective criteria, can be difficult and tedious. Semi-supervised clustering methods make this easier by letting the user provide must-link or cannot-link constraints. These are then used to automatically tune the similarity measure and/or the optimization criterion. In this paper, we investigate a complementary way of using the constraints: they are used to select an unsupervised clustering method and tune its hyperparameters. It turns out that this very simple approach outperforms all existing semi-supervised methods. This implies that choosing the right algorithm and hyperparameter values is more important than modifying an individual algorithm to take constraints into account. In addition, the proposed approach allows for active constraint selection in a more effective manner than other methods.

**Keywords** constraint-based clustering · algorithm and hyperparameter selection

## 1 Introduction

One of the core tasks in data mining is clustering: finding structure in data by identifying groups of instances that are highly similar (Jain 2010). We consider partitional clustering, in which every instance is assigned to exactly one cluster. To cluster data, a practitioner typically has to (1) define a similarity metric, (2) select a clustering algorithm, and (3) set the hyperparameters of that algorithm. A different choice in one of these steps usually leads to a different clustering. This

Toon Van Craenendonck
Department of Computer Science, KU Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
E-mail: toon.vancraenendonck@cs.kuleuven.be

Hendrik Blockeel
Department of Computer Science, KU Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
E-mail: hendrik.blockeel@cs.kuleuven.be

variability is desirable, as it allows users with different interests to find different clusterings of the same data. In search of a clustering that matches their interests, a typical practitioner might for example tweak the distance metric, or try different clustering algorithms, each with several hyperparameter settings. Though common, this strategy can also be quite tedious. Manually tweaking the distance metric is hard, and so is selecting and tuning a clustering algorithm. As a result, a user may go through several iterations of this clustering pipeline before arriving at a satisfactory result.

Semi-supervised clustering methods (Wagstaff et al. 2001; Xing et al. 2003; Bilenko et al. 2004) avoid the need for such iterations by explicitly incorporating user feedback into the clustering process. Instead of interacting with the clustering system by manually tweaking parts of the pipeline, the user provides feedback in a much more simple and well-defined way. Often, feedback is given in the form of pairwise constraints: the user indicates for several pairs whether they should be in the same cluster (called a must-link constraint), or not (a cannot-link constraint). Specifying such constraints is much easier than constructing a good distance metric, or selecting and tuning a clustering algorithm. The former relies on simple expressions of domain knowledge, whereas the latter requires extensive expertise on metrics, algorithm biases, and the influence of hyperparameters. It is thus assumed that the user already has some understanding of the data, or has some knowledge about it that is not directly captured by the features, and wants to find a clustering that respects this understanding. There are also other ways in which constraints can help in clustering: for example, one can use them to find clusterings that score better on a particular unsupervised optimization objective (e.g. they can help to obtain a lower with-cluster sum of squares (Ashtiani et al. 2016)). This, however, is only useful if the user knows a priori which objective is most suited for the task at hand.

Traditional approaches to semi-supervised (or constraint-based) clustering use constraints in one of the following three ways. First, one can modify an existing clustering algorithm to take them into account (Wagstaff et al. 2001; Ruiz et al. 2007; Rangapuram and Hein 2012; Wang et al. 2014). Second, one can learn a new distance metric based on the constraints, after which the metric is used within a traditional clustering algorithm (Xing et al. 2003; Bar-Hillel et al. 2003; Davis et al. 2007). Third, one can combine these two approaches and develop so-called hybrid methods (Bilenko et al. 2004; Basu et al. 2004). All of these methods aim to exploit the given background knowledge within the boundaries of a single clustering algorithm, and as such they ignore the algorithm and hyperparameter selection steps in the pipeline outlined above.

In contrast, we propose to use constraints for exactly this: to select and tune an unsupervised clustering algorithm. Our approach is motivated by the fact that no single algorithm works best on all clustering problems (Estivill-Castro 2002): each algorithm comes with its own bias, which may match a particular problem to a greater or lesser degree. Further, it exploits the ability of algorithms to produce very different clusterings depending on their hyperparameter settings.

Our proposed approach is simple: to find an appropriate clustering, we first generate a set of clusterings using several unsupervised algorithms, with different hyperparameter settings, and afterwards select from this set the clustering that satisfies the largest number of constraints. Our experiments show that, surprisingly, this simple constraint-based selection method often yields better clusterings

than existing semi-supervised methods. This leads to the key insight that it is more important to use an algorithm of which the inherent bias matches a particular problem, than to modify the optimization criterion of any individual algorithm to take the constraints into account.

Furthermore, we present a method for selecting the most informative constraints first, which further increases the usefulness of our approach. This selection strategy allows us to obtain good clusterings with fewer queries. Reducing the number of queries is important in many applications, as they are often answered by a user who has a limited time budget.

The remainder of this paper is structured as follows. In Section 2 we give some background on semi-supervised clustering, and algorithm and hyperparameter selection for clustering. Section 3 presents our approach to using pairwise constraints in clustering, which we call COBS (for Constraint-Based Selection). In Section 4 we describe how COBS can be extended to actively select informative constraints. We conclude in Section 5.

## 2 Background

**Semi-supervised clustering algorithms** allow the user to incorporate a limited amount of supervision into the clustering procedure. Several kinds of supervision have been proposed, one of the most popular ones being pairwise constraints. Must-link (ML) constraints indicate that two instances should be in the same cluster, cannot-link (CL) constraints that they should be in different clusters. Most existing semi-supervised approaches use such constraints within the scope of an individual clustering algorithm. COP-KMeans (Wagstaff et al. 2001), for example, modifies the clustering assignment step of K-means: instances are assigned to the closest cluster for which the assignment does not violate any constraints. Similarly, the clustering procedures of DBSCAN (Ruiz et al. 2007; Lelis and Sander 2009; Campello et al. 2013), EM (Shental et al. 2004) and spectral clustering (Rangapuram and Hein 2012; Wang et al. 2014) have been extended to incorporate pairwise constraints. Another approach to semi-supervised clustering is to learn a distance metric based on the constraints (Xing et al. 2003; Bar-Hillel et al. 2003; Davis et al. 2007). Xing et al. (2003), for example, propose to learn a Mahalanobis distance by solving a convex optimization problem in which the distance between instances with a must-link constraint between them is minimized, while simultaneously separating instances connected by a cannot-link constraint. Hybrid algorithms, such as MPCKMeans (Bilenko et al. 2004), combine metric learning with an adapted clustering procedure.

**Meta-learning and algorithm selection** have been studied extensively for supervised learning (Brazdil et al. 2003; Thornton et al. 2013), but much less for clustering. There is some work on building meta-learning systems that recommend clustering algorithms (Souto et al. 2008; Ferrari and Castro 2015). However, these systems do not take hyperparameter selection into account, or any form of supervision. More related to ours is the work of Caruana et al. (2006). They generate a large number of clusterings using K-means and spectral clustering, and cluster these clusterings. This meta-clustering is presented to the user as a dendrogram. Here, we also generate a set of clusterings, but afterwards we select from that set

the most suitable clustering based on pairwise constraints. The only other work, to our knowledge, that has explored the use of pairwise constraints for algorithm selection is that by Adam and Blockeel (2015). They define a meta-feature based on constraints, and use this feature to predict whether EM or spectral clustering will perform better for a dataset. While their meta-feature attempts to capture one specific property of the desired clusters, i.e., whether they overlap, our approach is more general and allows selection between any clustering algorithms.

Whereas algorithm selection has received little attention in clustering, several methods have been proposed for **hyperparameter selection**. One strategy is to run the algorithm with several parameter settings, and select the clustering that scores highest on an internal quality measure (Arbelaitz et al. 2013; Vendramin et al. 2010). Such measures try to capture the idea of a "good" clustering. A first limitation is that they are not able to deal with the inherent subjectivity of clustering, as they do not take any external information into account. Furthermore, internal measures are only applicable within the scope of individual clustering algorithms, as each of them comes with its own bias (von Luxburg et al. 2014). For example, the vast majority of them has a preference for spherical clusters, making them suitable for K-means, but not for e.g. spectral clustering and DBSCAN.

Another strategy for parameter selection in clustering is based on stability analysis (Ben-Hur et al. 2002; von Luxburg 2010; Ben-David et al. 2006). A parameter setting is considered to be stable if similar clusterings are produced with that setting when it is applied to several datasets from the same underlying model. These datasets can for example be obtained by taking subsamples of the original dataset (Ben-Hur et al. 2002; Lange et al. 2004). In contrast to internal quality measures, stability analysis does not require an explicit definition of what it means for a clustering to be good. Most studies on stability focus on selecting parameter settings in the scope of individual algorithms (in particular, often the number of clusters). As such, it is unclear to which extent stability can be used to choose between clusterings from very different clustering algorithms.

Additionally, one can also avoid the need for explicit parameter selection. In self-tuning spectral clustering (Zelnik-manor and Perona 2004), for example, the affinity matrix is constructed based on local statistics and the number of clusters is estimated using the structure of the eigenvectors of the Laplacian.

A key distinction with COBS is that none of the above methods takes the subjective preferences of the user into account. We will compare our constraint-based selection strategy to some of them in the next section.

Pourrajabi et al. (2014) have introduced CVCP, a framework for using constraints for hyperparameter selection within the scope of individual semi-supervised algorithms. A major difference is that COBS uses all constraints for selection (and none within the algorithm) and selects both an unsupervised algorithm and its hyperparameters (as opposed to only the hyperparameters of a semi-supervised algorithm). We compare COBS to CVCP in the experiments in section 3.

## 3 Constraint-based clustering selection

Algorithm and hyperparameter selection are difficult in an entirely unsupervised setting. This is mainly due to the lack of a well-defined way to estimate the quality

of clustering results (Estivill-Castro 2002). We propose to use constraints for this purpose, and estimate the quality of a clustering as the number of constraints that it satisfies. This quality estimate allows us to do a search over unsupervised algorithms and their parameter settings, as described in Algorithm 1.

Essentially, the algorithm simply generates a large set of clusterings, using various clustering algorithms and parameter settings, and then selects from this set the clustering that performs best according to the constraint-based criterion. In the algorithmic description, $c[i]$ indicates the cluster of element $i$ and $\mathbb{I}$ is the indicator function. For varying the clustering algorithm and hyperparameter settings, a simple grid search is used in our experiments. We have also experimented with SMAC (Hutter et al. 2011), which is based on sequential model-based optimization, but found the results highly similar to those obtained with grid search, both in terms of cluster quality and runtime. For completeness, we do include the SMAC results in the comparison to the semi-supervised competitors in section 3.5 (Figure 2). The objective that SMAC aims to minimize in these experiments is the number of unsatisfied constraints.

**Algorithm 1** Constraint-based selection (COBS)

**Input:** $D$: a dataset
$ML$: set of must-link constraints
$CL$: set of cannot-link constraints
**Output:** a clustering of $D$
1: Generate a set of clusterings $C$ by varying the hyperparameters of several unsupervised clustering algorithms
2: **return** $\arg\max_{c \in C} \left( \sum_{(i,j) \in ML} \mathbb{I}[c[i]=c[j]] + \sum_{(i,j) \in CL} \mathbb{I}[c[i] \neq c[j]] \right)$

We now reiterate and clarify the motivations for COBS, which were briefly presented in the introduction. First, each clustering algorithm comes with a particular bias, and no single one performs best on all clustering problems (Estivill-Castro 2002). Existing semi-supervised approaches can change the bias of an unsupervised algorithm, but only to a certain extent. For instance, using constraints to learn a Mahalanobis distance allows K-means to find ellipsoidal clusters, rather than spherical ones, but still does not make it possible to find non-convex clusters. In contrast, by using constraints to choose between clusterings generated by very different algorithms, COBS aims to select the most suitable one from a diverse range of biases.

Second, it is also widely known that within a single clustering algorithm the choice of the hyperparameters can strongly influence the clustering result. Consequently, choosing a good parameter setting is crucial. Currently, a user can either do this manually, or use one of the selection strategies discussed in section 2. Both options come with significant drawbacks. Doing parameter tuning manually is time-consuming, given the often large number of combinations one might try. Existing automated selection strategies avoid this manual labor, but can easily fail to select a good setting as they do not take the user's preferences into account. For COBS, parameters are an asset rather than a burden. They allow generating a large and diverse set of clusterings, from which we can select the most suitable solution with a limited number of pairwise constraints.

### 3.1 Research questions

In the remainder of this section, we aim to answer the following questions:

**Q1** How does COBS' hyperparameter selection compare to unsupervised hyperparameter selection?
**Q2** How does unsupervised clustering with COBS' hyperparameter selection compare to semi-supervised clustering algorithms?
**Q3** How does COBS, with *both algorithm and hyperparameter selection*, compare to existing semi-supervised algorithms?
**Q4** Can we combine the best of both worlds, that is: use COBS with semi-supervised clustering algorithms rather than unsupervised ones?
**Q5** How do the clusterings that COBS selects score on internal quality criteria?
**Q6** What is the computational cost of COBS, compared to the alternatives?

Although our selection strategy is also related to meta-clustering (Caruana et al. 2006), an experimental comparison would be difficult as meta-clustering produces a dendrogram of clusterings for the user to explore. The user can traverse this dendrogram to obtain a single clustering, but the outcome of this process is highly subjective. COBS works with pairwise constraints, therefore we compare to other methods that do the same.

### 3.2 Experimental methodology

To answer our research questions we perform experiments with artificial datasets as well as UCI classification datasets. The classes are assumed to represent the clusters of interest. We evaluate how well the returned clusters coincide with them by computing the Adjusted Rand Index (ARI) (Hubert and Arabie 1985), which is a commonly used measure for this. The ARI is an extension of the regular Rand index (RI), which is the ratio of the number of pairs of instances on which two clusterings agree (i.e. pairs that belong to the same cluster in both clusterings, or to a different cluster in both clusterings) to the total numer of pairs. The ARI corrects the RI for chance; whereas the expected value for random clusterings is not constant for the regular RI, it is 0 for the ARI. A clustering that coincides perfectly with the one indicated by the class labels has an ARI of 1, whereas a clustering that is generated randomly should have an ARI close to 0. In computing the ARI for clusterings generated by DBSCAN and FOSC we consider each point identified as noise to be in its own separate cluster[1]. We use 5-fold cross-validation, and in each iteration perform the following steps:

1. Generate $c$ pairwise constraints ($c$ is a parameter) by repeatedly selecting two random instances from the training set, and adding a must-link constraint if they belong to the same class, and a cannot-link otherwise.
2. Apply COBS to the full dataset to obtain a clustering.

[1] One could argue that these points are not really separate clusters, but as all the evaluation criteria and quality indices assume a complete partitioning of the data, they need to be taken into account somehow, either as separate clusters or as part of a single "noise cluster". The former seems most natural, but we also experimented with the latter. This did not affect any of our conclusions.

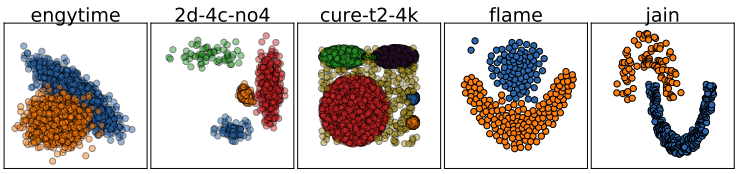



Fig. 1: Artificial datasets used in the experiments.

3. Evaluate the clustering by calculating the ARI on all objects from the validation fold.

In all plots and tables, we report the average ARI over the 5 validation folds. Measuring how well clusterings agree with given class labels is the most common way of evaluating semi-supervised clustering algorithms, despite its limitations (Färber et al. 2010; von Luxburg et al. 2014). One such limitation is that class labels do not always correspond to what one would commonly identify as cluster structure. We partly get around these limitations by also using artificial datasets (for which we know that the labels indicate actual clustering structure), and by investigating the performance of COBS on two internal quality measures (in research question 5).

*Datasets*

An overview of all the datasets can be found in Table 1. First, we perform experiments with the 5 artificial datasets [2] that are shown in Figure 1. The advantage of these artificial datasets is that they have unambiguous ground truth clusterings, and that they can easily be visualized (some illustrations of clusterings can be found in section 3.5). Further, we perform experiments with 14 classification datasets that have also been used in several other studies on semi-supervised clustering. The optdigits389 dataset is a subset of the UCI handwritten digits dataset, containing only digits 3, 8 and 9 (Bilenko et al. 2004; Mallapragada et al. 2008). The faces dataset contains 624 images of 20 persons taking different poses, with different expressions, with and without sunglasses. Hence, this dataset has 4 target clusterings: identity, pose, expression and eyes (whether or not the person is wearing sunglasses). We extract a 2048-value feature vector for each image by running it through the pre-trained Inception-V3 network (Szegedy et al. 2015) and storing the output of the second last layer. We only show results for the identity and eyes target clusterings, as the clusterings generated for pose and expression had an ARI close to zero for all methods. We experiment with two clustering tasks on the 20 newsgroups text data: in the first task the data consists of all documents from three related topics (comp.graphics, comp.os.ms-windows and comp.windows.x, as in (Basu and Mooney 2004; Mallapragada et al. 2008)), in the second task the data consists of all documents from three very different topics (alt.atheism, rec.sport.baseball and sci.space, as in (Basu and Mooney 2004; Mallapragada et al. 2008)). We first extract tf-idf features, and then use latent semantic indexing (as in (Mallapragada et al. 2008)) to reduce the dimensionality to 10. Duplicate

[2] These datasets are obtained from https://github.com/deric/clustering-benchmark

Table 1: Datasets used in the experiments. Duplicate instances and instances with missing values are removed.

| dataset | # instances | # features | # classes |
|---|---|---|---|
| 2d-4c-no4 | 863 | 2 | 4 |
| cure-t2-4k | 4200 | 2 | 7 |
| engytime | 4096 | 2 | 2 |
| jain | 373 | 2 | 2 |
| flame | 240 | 2 | 2 |
| wine | 178 | 13 | 3 |
| dermatology | 358 | 33 | 6 |
| iris | 147 | 4 | 3 |
| ionosphere | 350 | 34 | 2 |
| breast-cancer-wisconsin | 449 | 32 | 2 |
| ecoli | 336 | 7 | 8 |
| optdigits389 | 1151 | 64 | 3 |
| segmentation | 2100 | 19 | 7 |
| glass | 214 | 10 | 7 |
| hepatitis | 112 | 19 | 2 |
| parkinsons | 195 | 13 | 2 |
| column_2C | 310 | 6 | 2 |
| faces identity | 624 | 2048 | 20 |
| faces eyes | 624 | 2048 | 2 |
| newsgroups sim3 | 2946 | 10 | 3 |
| newsgroups diff3 | 2780 | 10 | 3 |

instances were removed for all these datasets, and all data was normalized between 0 and 1 (except for the artificial data, and the image and text data).

*Unsupervised algorithms used in COBS*

We use K-means, DBSCAN and spectral clustering to generate clusterings in step one of Algorithm 1, as they are common representatives of different types of algorithms (we use implementations from scikit-learn (Pedregosa et al. 2011)). The hyperparameters are varied in the ranges specified in Table 2. In particular, for each dataset we generate 180 clusterings using K-means (for each number of clusters we store the clusterings obtained with 20 random initializations), 351 using spectral clustering and 400 using DBSCAN, yielding a total of 931 clusterings. For discrete parameters, clusterings are generated for the complete range. For continuous parameters, clusterings are generated using 20 evenly spaced values in the specified intervals. For the $\epsilon$ parameter used in DBSCAN, the lower and upper bounds are the minimum and maximum pairwise distances between instances (referred to as $\min(d)$ and $\max(d)$ in Table 2). We use the Euclidean distance for all unsupervised algorithms.

### 3.3 Question Q1: COBS vs. unsupervised hyperparameter tuning

To evaluate hyperparameter selection for individual algorithms, we use Algorithm 1 with $C$ a set of clusterings generated using one particular algorithm (K-means, DBSCAN or spectral). We compare COBS to state of the art unsupervised selection strategies. As there is no single method that can be used for all three algorithms, we use three different approaches, which are briefly described next.

Table 2: Algorithms used, the hyperparameters that were varied, their corresponding ranges and the hyperparameter selection methods used in Q1. For *faces identity* the number of clusters was varied in $[2, 30]$ instead of $[2, 10]$, as it contains 20 classes.

| **Algorithm** | **Param.** | **Range** | **Selection method** |
|---|---|---|---|
| k-means | $K$ | $[2, 10]$ or $[2, 30]$ | silhouette index |
| DBSCAN | $\epsilon$<br>$minPts$ | $[\min(d), \max(d)]$<br>$[2, 20]$ | DBCV index |
| spectral | $K$<br>$k$<br>$\sigma$ | $[2, 10]$ or $[2, 30]$<br>$[2, 20]$<br>$[0.01, 5.0]$ | self-tuning<br>spectral clustering |

### *3.3.1 Existing unsupervised hyperparameter selection methods*

**K-means** has one hyperparameter: the number of clusters $K$. A popular method to select $K$ in K-means is by using internal clustering quality measures (Vendramin et al. 2010; Arbelaitz et al. 2013). K-means is run for different values of K (and in this case also for different random seeds), and afterwards the clustering that scores highest on such an internal measure is chosen. In our setup, we generate 20 clusterings for each $K$ by using different random seeds. We select the clustering that scores highest on the silhouette index (Rousseeuw 1987), which was identified as one of the best internal criteria by Arbelaitz et al. (2013).

**DBSCAN** has two parameters: $\epsilon$, which specifies how close points should be to be in the same neighborhood, and $minPts$, which specifies the number of points that are required in the neighborhood to be a core point. Most internal criteria are not suited for DBSCAN, as they assume spherical clusters, and one of the key characteristics of DBSCAN is that it can find clusters with arbitrary shape. One exception is the Density-Based Cluster Validation (DBCV) score (Moulavi et al. 2014), which we use in our experiments.

**Spectral clustering** requires the construction of a similarity graph, which can be done in several ways (Luxburg 2007). If a $k$-nearest neighbor graph is used, $k$ has to be set. For graphs based on a Gaussian similarity function, $\sigma$ has to be set to specify the width of the neighborhoods. Also the number of clusters $K$ should be specified. Self-tuning spectral clustering (Zelnik-manor and Perona 2004) avoids having to specify any of these parameters, by relying on local statistics to compute different $\sigma$ values for each instance, and by exploiting structure in the eigenvectors to determine the number of clusters. This approach is different from the one used for K-means and DBSCAN, as here we do not generate a set of clusterings first, but instead hyperparameters are estimated directly from the data.

### *3.3.2 Results and conclusion*

The columns of Table 3 marked with *Q1* compare the ARIs obtained with the unsupervised approaches to those obtained with COBS. The best of these two is underlined for each algorithm and dataset combination. Most of the times the constraint-based selection strategy performs better, and often by a large margin. Note for example the large difference for ionosphere: DBSCAN is able to produce a good clustering, but only the constraint-based approach recognizes it as the best one. When the unsupervised selection method performs better, the difference is

Table 3: We first show the ARIs obtained with unsupervised vs. constraint-based hyperparameter selection (columns marked Q1). Next, we show the ARIs obtained with the semi-supervised variants, with several hyperparameter selection methods (columns marked Q2). For semi-supervised results 50 constraints were used, and the average over 5 cross-validation folds is shown. SI refers to the silhouette index, STS to self-tuning spectral clustering, FOSC to FOSC-OpticsDend and eigen to the eigengap method.

| **dataset** | **K-means** *Q1* | | **MPCKMeans** *Q2* | | | **DBSCAN** *Q1* | | **FOSC** *Q2* | **spectral** *Q1* | | **COSC** *Q2* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SI | COBS | SI | NSat | CVCP | DBCV | COBS | | STS | COBS | eigen | NSat | CVCP |
| 2d-4c-no4 | **0.97** | 0.93 | 0.82 | 0.74 | 0.78 | 0.38 | 0.38 | **0.99** | **0.99** | **0.99** | **0.99** | 0.69 | 0.91 |
| cure-t2-4k | 0.45 | **0.56** | 0.53 | 0.33 | 0.35 | 0.00 | 0.27 | **0.81** | 0.75 | **0.86** | 0.43 | 0.42 | 0.51 |
| engytime | 0.56 | **0.81** | 0.56 | 0.49 | 0.55 | 0.00 | 0.00 | **0.38** | 0.85 | 0.81 | **0.86** | **0.86** | 0.76 |
| jain | 0.22 | 0.50 | 0.46 | 0.54 | **0.61** | 0.87 | 0.87 | **0.94** | 0.23 | **1.0** | **1.0** | **1.0** | 0.94 |
| flame | 0.44 | 0.32 | 0.10 | **0.68** | 0.43 | 0.18 | 0.91 | **0.98** | 0.48 | 0.87 | **0.95** | **0.95** | **0.95** |
| wine | **0.86** | 0.85 | 0.81 | 0.72 | 0.69 | 0.30 | 0.32 | **0.53** | 0.89 | **0.93** | 0.54 | 0.54 | 0.75 |
| dermatology | 0.56 | **0.81** | 0.59 | 0.39 | 0.36 | 0.27 | 0.34 | **0.81** | 0.20 | **0.92** | 0.36 | 0.36 | 0.58 |
| iris | 0.55 | 0.69 | 0.67 | **0.81** | 0.54 | 0.54 | 0.43 | **0.80** | 0.55 | 0.78 | **0.90** | 0.39 | 0.68 |
| ionosphere | **0.28** | 0.23 | 0.21 | 0.20 | 0.18 | 0.05 | **0.49** | -0.02 | **0.28** | 0.19 | 0.26 | 0.26 | 0.20 |
| breast-cancer | **0.73** | **0.73** | **0.73** | 0.71 | **0.73** | 0.29 | 0.32 | **0.53** | **0.81** | 0.78 | 0.76 | 0.76 | 0.76 |
| ecoli | 0.04 | 0.60 | **0.70** | 0.57 | 0.57 | 0.06 | 0.51 | **0.54** | 0.04 | **0.69** | 0.60 | 0.41 | 0.57 |
| optdigits389 | 0.49 | **0.79** | 0.58 | 0.24 | 0.54 | 0.00 | 0.30 | **0.53** | 0.38 | **0.96** | 0.54 | 0.54 | 0.85 |
| segmentation | 0.10 | **0.59** | 0.43 | 0.22 | 0.33 | 0.24 | 0.49 | **0.61** | 0.23 | **0.58** | 0.19 | 0.19 | 0.27 |
| hepatitis | 0.25 | **0.34** | 0.25 | 0.23 | 0.25 | 0.02 | 0.08 | **0.23** | -0.08 | -0.05 | **0.20** | **0.20** | **0.20** |
| glass | 0.26 | 0.25 | 0.31 | 0.28 | **0.32** | 0.01 | 0.19 | **0.32** | **0.23** | **0.23** | 0.11 | 0.11 | 0.16 |
| parkinsons | -0.07 | -0.03 | -0.04 | **0.01** | -0.02 | **-0.02** | -0.03 | -0.10 | 0.09 | 0.12 | **0.16** | 0.12 | 0.07 |
| column_2C | 0.17 | 0.13 | **0.22** | 0.17 | 0.22 | **0.02** | -0.02 | -0.03 | 0.17 | 0.10 | 0.22 | 0.22 | **0.27** |
| faces identity | 0.66 | 0.40 | **0.71** | 0.03 | 0.15 | 0.00 | 0.49 | **0.65** | 0.23 | **0.65** | 0.07 | 0.07 | 0.25 |
| faces eyes | 0.15 | 0.59 | 0.08 | **0.60** | 0.55 | **0.00** | **0.00** | **0.00** | 0.13 | **0.49** | 0.01 | 0.01 | 0.00 |
| news sim3 | 0.12 | **0.18** | 0.12 | 0.04 | 0.08 | 0.00 | **0.02** | **0.02** | 0.00 | 0.13 | 0.09 | 0.00 | **0.14** |
| news diff3 | 0.28 | **0.56** | 0.25 | 0.23 | 0.34 | 0.00 | 0.24 | **0.48** | 0.00 | **0.56** | 0.15 | 0.00 | 0.14 |

usually small. *We conclude that often the internal measures do not match the clusterings that are indicated by the class labels. Constraints provide useful information that can help select a good parameter setting.*

## 3.4 Question Q2: COBS vs. semi-supervised algorithms

It is not too surprising that COBS outperforms unsupervised hyperparameter selection, since it has access to more information. We now compare to semi-supervised algorithms, which have access to the same information.

### *3.4.1 Existing semi-supervised algorithms*

We compare to the following algorithms, as they are semi-supervised variants of the unsupervised algorithms used in our experiments:

– **MPCKMeans** (Bilenko et al. 2004) is a hybrid semi-supervised extension of K-means. It minimizes an objective that combines the within-cluster sum of squares with the cost of violating constraints. This objective is greedily minimized using a procedure based on K-means. Besides a modified cluster assignment step and the usual cluster center re-estimation step, this procedure also adapts an individual metric associated with each cluster in each iteration.

We use the implementation available in the WekaUT package[3].

- **FOSC-OpticsDend** (Campello et al. 2013) is a semi-supervised extension of OPTICS, which is in turn based on ideas similar to DBSCAN. The first step of this algorithm is to run the unsupervised OPTICS algorithm, and to construct a dendrogram using its output. The FOSC framework is then used to extract a flat clustering from this dendrogram that is optimal w.r.t. the given constraints.
- **COSC** (Rangapuram and Hein 2012) is based on spectral clustering, but optimizes for an objective that combines the normalized cut with a penalty for constraint violation. We use the implementation available on the authors' web page[4].

In our experiments, the only kind of supervision that is given to the algorithms is in the form of pairwise constraints. In particular, the number of clusters $K$ is assumed to be unknown. In COBS, $K$ is treated as any other hyperparameter. MPCKMeans and COSC, however, require specifying the number of clusters. The following strategies are used to select $K$ based on the constraints:

- **NSat**: We run the algorithms for multiple $K$, and select the clustering that violates the smallest number of constraints. In case of a tie, we choose the solution with the lowest number of clusters.
- **CVCP**: Cross-Validation for finding Clustering Parameters (Pourrajabi et al. 2014) is a cross-validation procedure for semi-supervised clustering. The set of constraints is divided into $n$ independent folds. To evaluate a parameter setting, the algorithm is repeatedly run on the entire dataset given the constraints in $n-1$ folds, keeping aside the $n$th fold as a test set. The clustering that is produced given the constraints in the $n-1$ folds, is then considered as a classifier that distinguishes between must-link and cannot-link constraints in the $n$th fold. The F-measure is used to evaluate the score of this classifier. The performance of the parameter setting is then estimated as the average F-measure over all test folds. This process is repeated for all parameter settings, and the one resulting in the highest average F-measure is retained. The algorithm is then run with this parameter setting using all constraints to produce the final clustering. We use 5-fold cross-validation.

We also compare to unsupervised hyperparameter selection strategies for the semi-supervised algorithms. In particular, we use the silhouette index for MPCK-Means, and the eigengap heuristic for COSC (Luxburg 2007). The affinity matrix for COSC is constructed using local scaling as in (Rangapuram and Hein 2012).

### *3.4.2 Results and conclusion*

The columns in Table 3 marked with *Q2* show the ARIs obtained with the semi-supervised algorithms. The best result for each type of algorithm (unsupervised or semi-supervised) is indicated in bold. The table shows that in several cases it is more advantageous to use the constraints to optimize the hyperparameters of the unsupervised algorithm (as COBS does). In other cases, it is better to use the constraints within the algorithm itself, to perform a more informed search (as the

[3] `http://www.cs.utexas.edu/users/ml/risc/code/`

[4] `http://www.ml.uni-saarland.de/code/cosc/cosc.htm`

semi-supervised variants do). *Within the scope of a single clustering algorithm, neither strategy consistently outperforms the other.* For example, if we use spectral clustering on the dermatology data, it is better to use the constraints for tuning the hyperparameters of unsupervised spectral clustering (also varying $k$ and $\sigma$ for constructing the signature matrix) than within COSC, its semi-supervised variant (which uses local scaling for this). In contrast, if we use density-based clustering on the same data, it is better to use constraints in FOSC-OpticsDend (which does not have an $\epsilon$ parameter, and for which $minPts$ is set to 4, a value commonly used in the literature (Ester et al. 1996; Campello et al. 2013)) than to use them to tune the hyperparameters of DBSCAN (varying both $\epsilon$ and $minPts$).

### 3.5 Question Q3: COBS with multiple unsupervised algorithms

In the previous two subsections, we showed that constraints can be useful to tune the hyperparameters of individual algorithms. Table 3 also shows, however, that no single algorithm (unsupervised or semi-supervised) performs well on all datasets. This motivates the use of COBS to not only select hyperparameters, but also the clustering algorithm. In this subsection we again use Algorithm 1, but set $C$ in step 1 now includes clusterings produced by any of the three unsupervised algorithms.

#### *3.5.1 Results*

We compare COBS with existing semi-supervised algorithms in Figure 2. For the majority of datasets, COBS produces clusterings that are on par with, or better than, those produced by the best competitor. While some other approaches also do well on some of the datasets, none of them do so consistently. Compared to each competitor individually, COBS is clearly superior. For example, COSC-NumSat outperforms COBS on the breast-cancer-wisconsin dataset, but performs much worse on several others. The only datasets for which COBS performs significantly worse than its competitors are column_2C and hepatitis.

Table 4 allows us to assess the quality of the clusterings that are selected by COBS, relative to the quality of the best clustering in the set of generated clusterings. Column 2 shows the highest ARI of all generated clusterings for each dataset. Note that we can only compute this value in an experimental setting, in which we have labels for all elements. In a real clustering application, we cannot simply select the result with the highest ARI. Column 3, then, shows the ARI of the clustering that is actually selected using COBS when it is given 50 constraints. It shows that there still is room for improvement, i.e. a more advanced strategy might get closer to the maxima. Nevertheless, even our simple strategy gets close enough to outperform most other semi-supervised methods. The last column of Table 4 shows how often COBS chose a clustering by K-means ('K'), DBSCAN ('D') and spectral clustering ('S'). It illustrates that the selected algorithm strongly depends on the dataset. For example, for ionosphere COBS selects clusterings generated by DBSCAN, as it is the only algorithm able to produce good clusterings of this dataset.

The clusterings that COBS selects for hepatitis are also generated by DBSCAN. This might seem strange as the ARIs of these DBSCAN clusterings are low (and significantly lower than those of the K-means clusterings, as can be seen in Table 3).

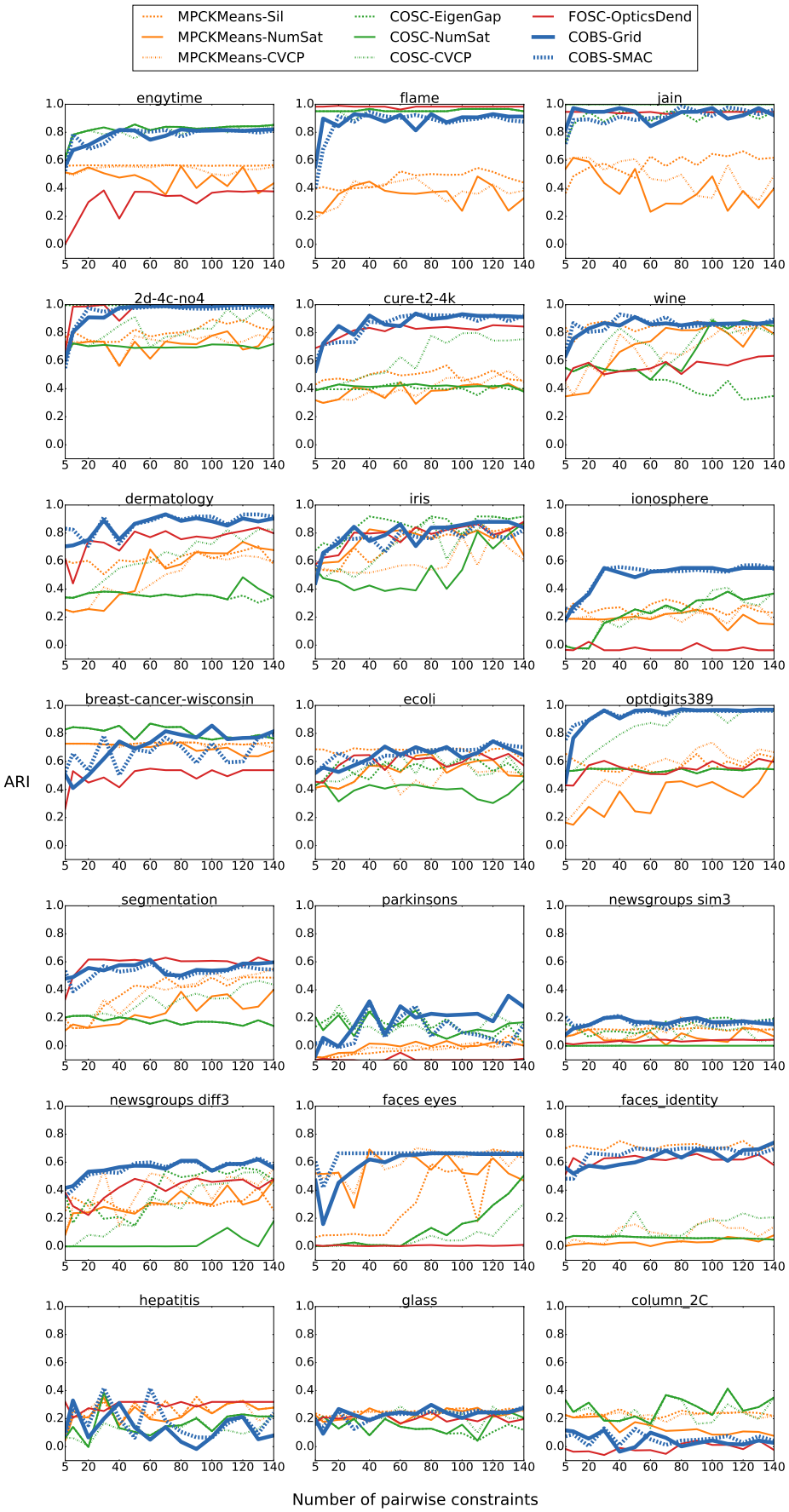



Fig. 2: Performance of COBS vs. semi-supervised algorithms. COBS-Grid shows the performance of COBS with a grid search (as used in the remainder of the paper), COBS-SMAC shows the performance of SMAC algorithm and hyperparameter selection (Hutter et al. 2011).

In other words, the DBSCAN clusterings are good at satisfying the constraints (as they are selected), but at the same time do not recover the class structure that the constraints are derived from. To understand this, note that most of the constraints for hepatitis are must-links (it consists of 2 classes, one with 19 instances and the other with 93). DBSCAN generates some clusterings in which (nearly) all instances are in one cluster. These clusterings are good at satisfying the randomly generated constraints (most of them are must-links, and DBSCAN is right about them as there is only one cluster), and are chosen by COBS. The selected clusterings, however, score badly on the Adjusted Rand Index (ARI) as this measure is adjusted for chance and takes class imbalance into account. In other words, this behaviour is due to a discrepancy between the selection criterion of COBS, and the ARI. This observation indicates that COBS might benefit from the development of more complex selection criteria.

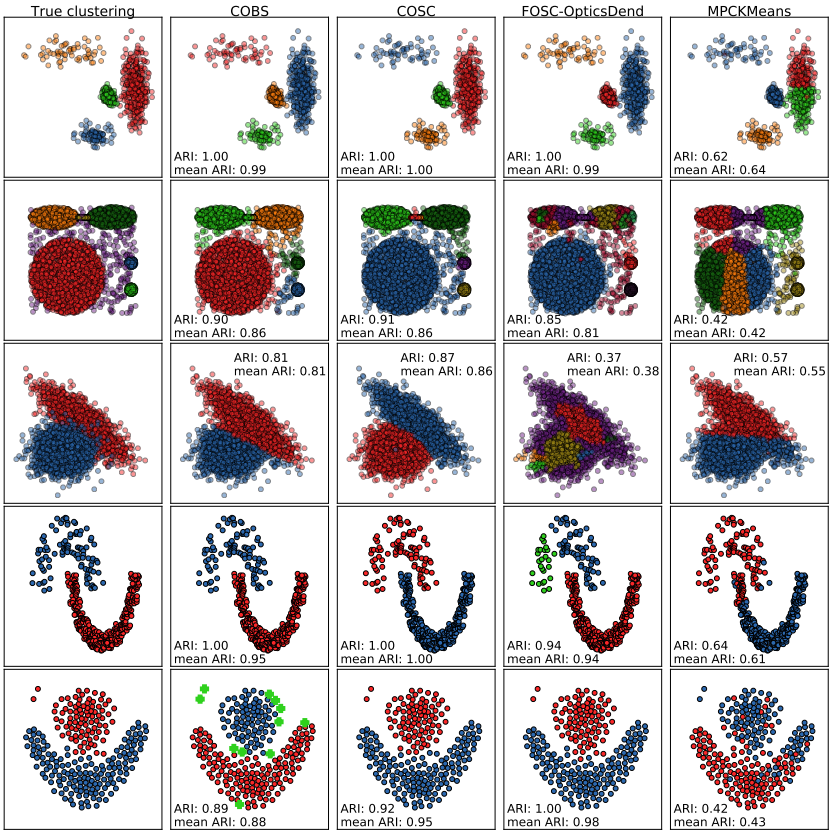



Fig. 3: Comparing COBS to other semi-supervised approaches on artificial datasets given 50 constraints. The plots show the clustering with the median ARI over 5 cross-validation folds. For this plot, the correct number of clusters was given to COSC and MPCKMeans.

Table 4: The ARI of the best clustering that is generated by any of the unsupervised algorithms, the ARI of the clustering that is selected after 50 constraints (averaged over 5 cross-validation folds), and the algorithms that produced the selected clusterings.

| dataset | best unsupervised | COBS | algorithm used |
|---|---|---|---|
| 2d-4c-no4 | 1.0 | 0.99 | K:1/D:0/S:4 |
| cure-t2-4k | 0.94 | 0.86 | K:0/D:0/S:5 |
| engytime | 0.82 | 0.81 | K:1/D:0/S:4 |
| jain | 1.0 | 0.95 | K:0/D:4/S:1 |
| flame | 0.93 | 0.88 | K:0/D:2/S:3 |
| wine | 0.93 | 0.90 | K:0/D:0/S:5 |
| dermatology | 0.94 | 0.86 | K:2/D:0/S:3 |
| iris | 0.88 | 0.78 | K:0/D:0/S:5 |
| ionosphere | 0.56 | 0.48 | K:0/D:5/S:0 |
| breast-cancer-wisconsin | 0.84 | 0.69 | K:0/D:1/S:4 |
| ecoli | 0.75 | 0.71 | K:1/D:0/S:4 |
| optdigits389 | 0.97 | 0.95 | K:0/D:0/S:5 |
| segmentation | 0.61 | 0.57 | K:0/D:0/S:5 |
| hepatitis | 0.32 | 0.08 | K:0/D:5/S:0 |
| glass | 0.33 | 0.23 | K:1/D:0/S:4 |
| parkinsons | 0.34 | 0.08 | K:0/D:3/S:2 |
| column_2C | 0.27 | 0.05 | K:0/D:3/S:2 |
| faces identity | 0.80 | 0.49 | K:2/D:0/S:3 |
| faces eyes | 0.66 | 0.59 | K:5/D:0/S:0 |
| news sim3 | 0.28 | 0.17 | K:3/D:0/S:2 |
| news diff3 | 0.63 | 0.58 | K:2/D:0/S:3 |

*Results on artificial datasets*

Figure 3 shows the clusterings that are produced for the artificial datasets given 50 constraints. COBS performs on par with the best competitor for all of these. An interesting observation can be made for flame (shown in the last row). For this dataset, COBS selects a solution consisting of two clusters and 11 additional noise points (which are considered as separate clusters in computing the ARI). This clustering is produced by DBSCAN, which identifies the points shown as green crosses as noise. In this case, no constraints were defined on these points. The clustering that is shown satisfies all given constraints, and was selected randomly from all clusterings that did so. Giving the correct number of clusters (as was done for COSC and MPCKMeans for Figure 3) and not allowing noise points would result in COBS selecting clusterings that are highly similar to those generated by COSC and FOSC.

Besides COBS, also COSC attains high ARIs for all 5 artificial datasets, but only if it is given the correct number of clusters. Without giving it the right number of clusters, COSC produces much worse clusterings on cure-t2-4k, with ARIs of 0.43 (for the eigengap selection method), 0.42 (for NumSat) and 0.51 (for CVCP) (as listed in Table 3).

Another interesting observation can be made for MPCKMeans, for example on the flame dataset (shown in the last row). It shows that using constraints does not allow MPCKMeans to avoid its inherent spherical bias. Points that are connected by a must-link constraint are placed in the same cluster, but the overall cluster

shape cannot be correctly identified. This is seen in the plot, as for instance some red points occur in the inner cluster.

#### *3.5.2 Conclusion*

If any of the unsupervised algorithms is able to produce good clusterings, COBS can select them using a limited number of constraints. If not, COBS performs poorly, but in our experiments none of the algorithms did well in this case. *We conclude that it is generally better to use constraints to select and tune an unsupervised algorithm, than within a randomly chosen semi-supervised algorithm.*

### 3.6 Question Q4: Using COBS with semi-supervised algorithms

In the previous section we have shown that we can use constraints to do algorithm and hyperparameter selection for *unsupervised* algorithms. On the other hand, constraints can also be useful when used within an adapted clustering procedure, as traditional semi-supervised algorithms do. This raises the question: can we combine both approaches? In this section, we use the constraints to select and tune a *semi-supervised* clustering algorithm. In particular, we vary the hyperparameters of the semi-supervised algorithms to generate the set of clusterings from which we select. The varied hyperparameters are the same as those for their unsupervised variants, except for two. First, $\epsilon$ is not varied for FOSC-OpticsDend, as it is not a hyperparameter for that algorithm. Second, in this section we only use $k$-nearest neighbors graphs for (semi-supervised) spectral clustering, as full similarity graphs lead to long execution times for COSC.

#### *3.6.1 Results and conclusions*

Column 3 of Table 5 shows that this strategy does not produce better results. This is caused by using the same constraints twice: once within the semi-supervised algorithms, and once to evaluate the algorithms and select the best-performing one. Obviously, algorithms that overfit the given constraints will get selected in this manner.

The problem could be alleviated by using separate constraints inside the algorithm and for evaluation, but this decreases the number of constraints that can effectively be used for either purpose. Column 4 of Table 5 shows the average ARIs that are obtained if we use half of the constraints within the semi-supervised algorithms, and half to select one of the generated clusterings afterwards. This works better, but still often not as good as COBS with unsupervised algorithms.

*We conclude that using semi-supervised algorithms within COBS can only be beneficial if the semi-supervised algorithms use different constraints from those used for selection. Even then, when a limited number of constraints is available, using all of them for selection is often the best choice.*

### 3.7 Question Q5: Evaluating the clusterings on internal criteria

In the previous research questions we have investigated how well the clusterings produced by COBS score according to the ARI, an external evaluation measure.

Table 5: ARIs obtained with 50 constraints by COBS with unsupervised algorithms (COBS-U) and with semi-supervised algorithms, with and without splitting the constraint set (COBS-SS and COBS-SS-split). Results are averaged over 5 cross-validation folds.

| dataset | COBS-U | COBS-SS | COBS-SS-split |
|---|---|---|---|
| 2d-4c-no4 | 0.98 | 0.64 | **1.0** |
| cure-t2-4k | **0.86** | 0.39 | 0.83 |
| engytime | **0.80** | 0.15 | 0.78 |
| jain | **0.97** | 0.75 | **0.97** |
| flame | 0.91 | 0.45 | **0.93** |
| wine | **0.86** | 0.66 | **0.86** |
| dermatology | **0.89** | 0.74 | 0.77 |
| iris | **0.78** | 0.73 | 0.73 |
| ionosphere | **0.53** | 0.26 | 0.19 |
| breast-cancer-wisconsin | **0.69** | 0.59 | 0.68 |
| ecoli | **0.69** | 0.46 | 0.64 |
| optdigits389 | **0.96** | 0.58 | 0.93 |
| segmentation | 0.55 | 0.37 | **0.64** |
| hepatitis | 0.08 | 0.14 | **0.27** |
| glass | 0.23 | 0.20 | **0.24** |
| parkinsons | 0.05 | 0.04 | **0.08** |
| column_2C | 0.02 | **0.24** | 0.03 |
| faces identity | 0.45 | 0.30 | **0.64** |
| faces eyes | **0.60** | 0.16 | 0.59 |
| news sim3 | **0.18** | 0.09 | 0.15 |
| news diff3 | **0.60** | 0.34 | 0.42 |

This is motivated by the fact that our main goal is to find clusterings that are aligned with the user's interests, which are assumed to be captured by the class labels. In this section we investigate how well these clusterings score on internal measures, which are computed using only the data. Such internal measures capture characteristics that one might expect of a good clustering, such as a high intra-cluster similarity and low inter-cluster similarity. In particular, we want to know to what extent we compromise on quality according to internal criteria, by focusing on satisfying constraints.

Ideally, we should use an internal measure that is not biased towards any particular clustering algorithm. However, this does not exist (Estivill-Castro 2002): each internal quality measure comes with its own bias, which may match the bias of a clustering algorithm to a greater or lesser degree. As a result, choosing a suitable internal quality criterion is often as difficult as choosing the right clustering algorithm. For example, the large majority of internal measures has a strong spherical bias (Vendramin et al. 2010; Arbelaitz et al. 2013), making them suited to use in combination with k-means, but not with spectral clustering and DBSCAN.

In this section, we will investigate the trade-off between the ARI and two internal measures: the silhouette index (SI) (Rousseeuw 1987) and the density-based cluster validation (DBCV) score (Moulavi et al. 2014), both of which were also used in answering the previous research questions. The SI was chosen as it is well-known, and the extensive studies by Arbelaitz et al. (2013) and Vendramin et al. (2010) identify it as one of the best performing measures. The DBCV score was chosen as it is one of the few internal measures that does not have a spherical bias, and instead is based on the within- and between-cluster density connectedness of clusters. Although it does not have a spherical bias, the DBCV score comes with

its own limitations; for example, it is strongly influenced by noise, and biased towards imbalanced clusterings (Van Craenendonck and Blockeel 2015). Both of them range in $[-1, 1]$, with higher values being better.

Figure 4 shows how well the semi-supervised methods score on the internal measures for six datasets. In most cases, COBS performs comparable to its competitors. A notable exception is the parkinsons dataset, for which FOSC-OpticsDend produces clusterings that score significantly higher on both the DBCV score. Interestingly, the ARI of these clusterings is near zero. For parkinsons, the clusterings with the highest ARI score low on the internal measures. This, however, does not necessarily imply that the clustering does not identify any inherent structure (although this can be the case), it only means that it does not identify structure as it is defined by the silhouette score (i.e. spherical structure) or the DBCV score (i.e. density structure).

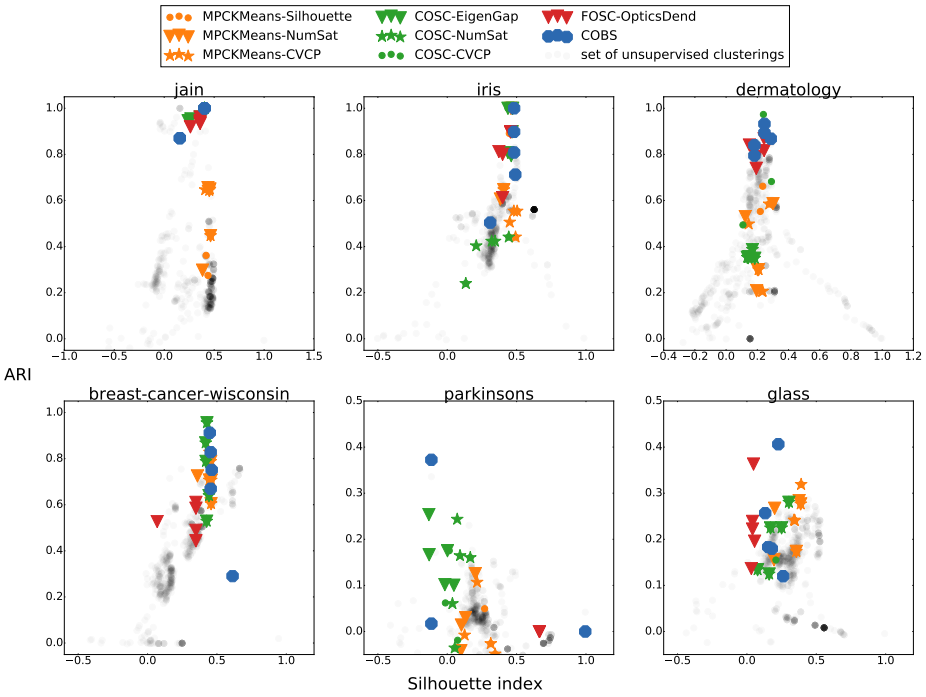
*We conclude that, most of the times, COBS performs comparable to its competitors on the silhouette and DBCV scores.* We also note that while on the one hand COBS selects a clustering solely on how well it satisfies a given set of constraints, on the other hand the clusterings from which it selects are all generated by an unsupervised algorithm that did not have access to these constraints, and hence are sensible according to the bias of at least the algorithm that generated them.

### 3.8 Question Q6: The computational cost of COBS

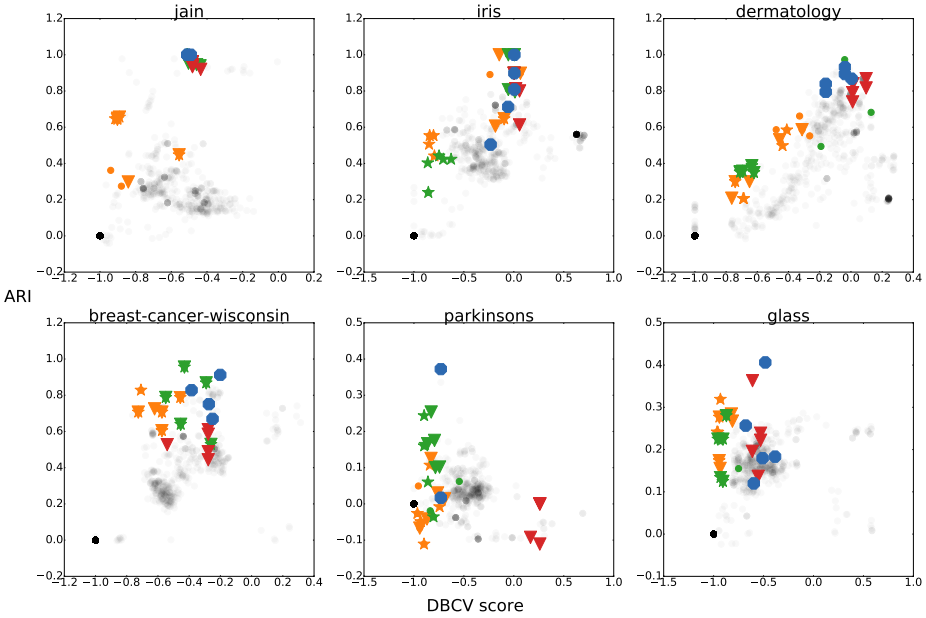
The computational cost of COBS depends on the complexity of the unsupervised algorithms that are used to generate clusterings. Generating all clusterings took the longest, by far, for the faces dataset. For the identity target, it took ca. 5 hours, due to the high dimensionality of the data and the many values of $K$ that were tried (it was varied in $[2, 30]$). For most datasets, generating all clusterings was much faster. As the semi-supervised algorithms can be significantly slower than their unsupervised counterparts, generating all unsupervised clusterings was in several cases faster than doing several runs of a semi-supervised algorithm (which is usually required, as the number of clusters is not known beforehand). This was especially so for COSC, which can be much slower than unsupervised spectral clustering. For example, generating all unsupervised clusterings for engytime took 50 minutes (using scikit-learn implementations), whereas only a single run of COSC took 28 minutes (using the Matlab implementation available on the authors' web page).

The runtime of COBS can be reduced in several ways. The cluster generation step can easily be parallelized. For larger datasets, one might consider doing the algorithm and hyperparameter selection on a sample of the data, and afterwards cluster the complete dataset only once with the selected configuration.

Finally, note that the added cost of doing algorithm and parameter selection is no different from its comparable, and commonly accepted, cost in supervised learning. The focus is on maximally exploiting the limited amount of supervision, as obtaining labels or constraints is often expensive, whereas computation is cheap.




(a) This plots show how well the semi-supervised methods score on the silhouette score, as well as how it correlates with the ARI. For each method, it shows the silhouette score and the ARI for each of 5 clusterings, generated in the 5-fold cross-validation. In black and transparent, it shows the silhouette scores and ARI's for all the clusterings that were generated by the unsupervised algorithms. Many of the clusterings that score high on the silhouette score but have a low ARI (e.g. for iris and dermatology) are generated by DBSCAN and consist of many small clusters (namely, the instances considered "noise").




(b) The same as (a), but for the DBCV instead of the silhouette score.

Fig. 4

## 4 Active COBS

Obtaining constraints can be costly, as they are often specified by human experts. Consequently, several methods have been proposed to actively select the most informative constraints (Basu and Mooney 2004; Mallapragada et al. 2008; Xiong et al. 2014). We first briefly discuss some of these methods, and subsequently present a constraint selection strategy for COBS.

### 4.1 Related work

Basu and Mooney (2004) were the first to propose an active constraint selection method for semi-supervised clustering. Their strategy is based on the construction of neighborhoods, which are points that are known to belong to the same cluster because must-link constraints are defined between them. These neighborhoods are initialized in the *exploration* phase: $K$ (the number of clusters) instances with cannot-link constraints between them are sought, by iteratively querying the relation between the existing neighborhoods and the point farthest from these neighborhoods. In the subsequent *consolidation* phase these neighborhoods are expanded by iteratively querying a random point against the known neighborhoods until a must-link occurs and the right neighborhood is found. Mallapragada et al. (2008) extend this strategy by selecting the most uncertain points to query in the consolidation phase, instead of random ones. Note that in these approaches all constraints are queried before the actual clustering is performed.
More recently, Xiong et al. (2014) proposed the normalized point-based uncertainty (NPU) framework. Like the approach introduced by Mallapragada et al. (2008), NPU incrementally expands neighborhoods and uses an uncertainty-based principle to determine which pairs to query. In the NPU framework, however, data is re-clustered several times, and at each iteration the current clustering is used to determine the next set of pairs to query. NPU can be used with any semi-supervised clustering algorithm, and Xiong et al. (2014) use it with MPCKMeans to experimentally demonstrate its superiority to the method of Mallapragada et al. (2008).

### 4.2 Active constraint selection in COBS

Like the approaches in (Mallapragada et al. 2008) and (Xiong et al. 2014), our constraint selection strategy for COBS is based on uncertainty sampling. Defining this uncertainty is straightforward within COBS, because of the availability of a set of clusterings: a pair is more uncertain if more clusterings disagree on whether it should be in the same cluster or not. Algorithm 2 presents a selection strategy based on this idea. We associate with each clustering $c$ a weight $w_c$ that depends on the number of constraints $c$ was right or wrong about. In each iteration we query the pair with the lowest weighted agreement. The agreement of a pair (line 5 of the algorithm) is defined as the absolute value of the difference between the sum of weights of clusterings in which the instances in the pair belong to the same cluster, and the sum of weights of clusterings in which they belong to a different cluster. The weights of clusterings that correctly "predict" the relation between pairs are

increased by multiplying with an update factor $m$, weights of other clusterings are decreased by dividing by $m$. As the total number of pairwise constraints is quite large ($\binom{n}{2}$ with $n$ the number of instances), we only consider constraints in a small random sample $P$ of all possible constraints.

**Algorithm 2** Active constraint selection for COBS

**Input:** $D$: a dataset
*budget*: the maximum number of constraints to use
$m$: weight update factor
$s$: size of sample of constraints to choose from
**Output:** a clustering of $D$
1: Generate $C$ a set of $s$ clusterings by varying the hyperparameters of several unsupervised clustering algorithms
2: Let $w_c = \frac{1}{s}$ for all $c \in C$
3: Let $P$ be a sample of all possible pairwise constraints
4: **while** $u < budget$ **do**
5: $(i, j) \leftarrow \arg\min_{(p,q)\in P} \left| \sum_{c\in C} \mathbb{I}[c[p]=c[q]] w_c - \sum_{c\in C} \mathbb{I}[c[p]\neq c[q]] w_c \right|$
6: Query pair $(i, j)$
7: $\forall c \in C$: multiply $w_c$ with $m$ if $c$ correctly predicted the relation between $i$ and $j$, divide by $m$ if not
8: $u \leftarrow u + 1$
9: **end while**
10: **return** the clustering with the highest weight

### 4.3 Experiments

We first demonstrate the influence of the weight update factor and sample size, and then compare our approach to active constraint selection with NPU (Xiong et al. 2014).

*Effect of weight update factor and sample size*

Our constraint selection strategy requires specifying a weight update factor $m$ and a sample size $s$. Figure 5 shows the effect of these two parameters for the wine and dermatology datasets. First, the figure shows that the active strategy can significantly improve performance over random selection when only few constraints are used. For example, given five constraints the random selection strategy on average chooses a clustering with an ARI of ca. 0.67, whereas the active strategy on average selects a clustering with an ARI of ca. 0.80 (for $s$=200 and $m$=2). A similar boost in ARI is observed for dermatology. Second, the figure shows that smaller sample sizes tend to give better, and more stable, results. This can be explained by the occasional domination of poor quality clusterings in the selection process: if there are more pairs to choose from, poor clusterings (which may have gotten lucky on the first few queries) have more opportunity to favor the search in their direction. This phenomenon is worse for large update factors, which can be seen by comparing the performance for $m$=1.05 and $m$=2 on the dermatology data, for a sample size of $s$=2000.

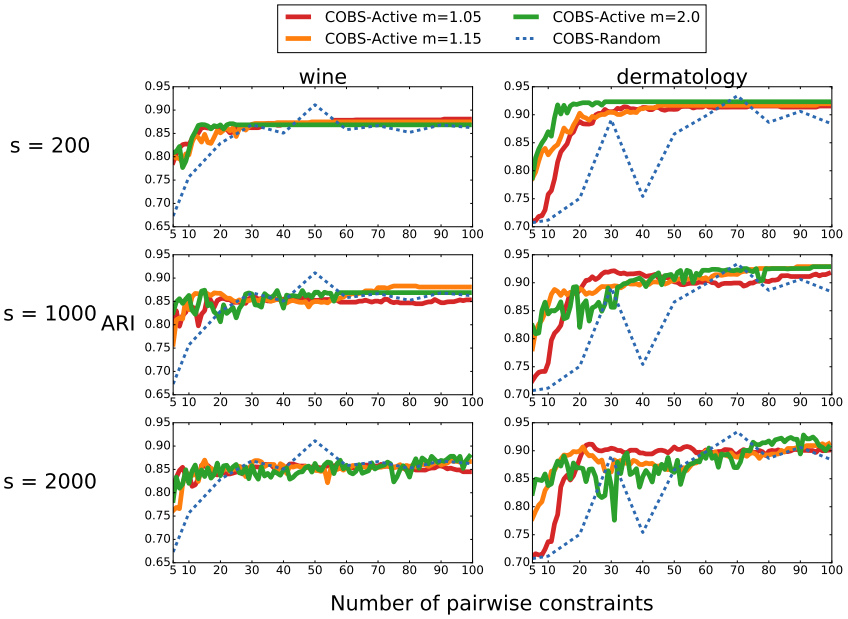



Fig. 5: Active COBS with different weight update factors and sample sizes. Each curve corresponds to a different weight update factor $m$. Each row in the figure corresponds to a different sample size $s$. Results are averaged over 8 randomly drawn constraint sample sets.

In the remainder of this section we use a sample of 200 constraints (i.e. we try to choose the most useful constraints to ask from 200 possible queries), and set the weight update factor to 2.

### *4.3.1 Comparison to active selection with NPU*

NPU (Xiong et al. 2014) can be used in combination with any semi-supervised clustering algorithm, we use the same ones as in the previous section. We do not include CVCP hyperparameter selection in these experiments, because of its high computational complexity (for these experiments we cannot cluster for several fixed numbers of constraints, as the choice of the next constraints depends on the current clustering). For the same reason we only include the EigenGap parameter selection method for the two largest datasets (opdigits389 and segmentation) in these experiments. The results are shown in Figure 6. For the first 8 datasets, the conclusions are similar to those for the random setting: COBS consistently performs well. Also in the active setting, none of the approaches produces a clustering with a high ARI for glass. For hepatitis, however, MPCKMeans is able to find good clusterings while COBS is not, albeit only after a relatively large number of constraints (hepatitis contains 112 instances). This implies that, although the labels might not represent a natural grouping, the class structure does match the bias of MPCKMeans, and given many constraints the algorithm finds this structure.

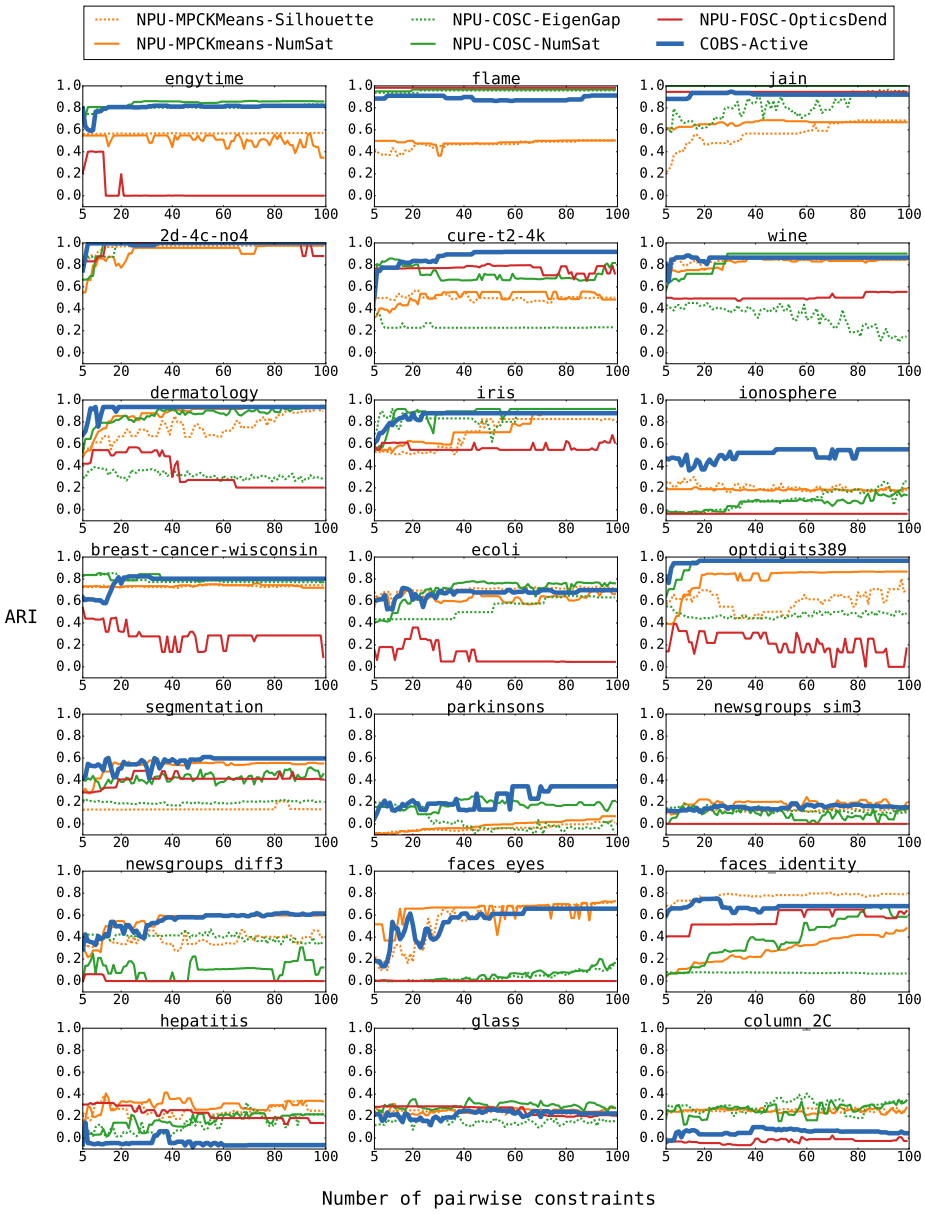



Fig. 6: Comparison of active COBS to NPU in combination with different semi-supervised clustering algorithms

#### *4.3.2 Time complexity*

We distinguish between the offline and online stages of COBS. In the offline stage, the set of clusterings is generated. As mentioned before, this took up to ca. 5 hours (for the faces dataset). In the online stage, we select the most informative pairs and ask the user about their relation. Execution time is particularly important here, as this stage requires user interaction. In active COBS, selecting the next pair to query is $\mathcal{O}(|C||P|)$, as we have to loop through all clusterings ($|C|$) for each constraint in the sample ($|P|$). For the setup used in our experiments ($|C|$=931, $|P|$=200), this was always less than 0.02s. Note that this time does not depend on the size of the dataset (as all clusterings are generated beforehand). In contrast, NPU requires re-clustering the data several times during the constraint selection process, which is usually significantly more expensive. This means that if NPU is used in combination with an expensive algorithm, e.g. COSC, the user has to wait longer between questions.

#### *4.3.3 Conclusion*

The COBS approach allows for a straightforward definition of uncertainty: pairs of instances are more uncertain if more clusterings disagree on them. Selecting the most uncertain pairs first can significantly increase performance.

## 5 Conclusion

Exploiting constraints has been the subject of substantial research, but all existing methods use them within the clustering process of individual algorithms. In contrast, we propose to use them to choose between clusterings generated by different unsupervised algorithms, ran with different parameter settings. We experimentally show that this strategy is superior to all the semi-supervised algorithms compared to, which themselves are state of the art and representative for a wide range of algorithms. For the majority of the datasets, it works as well as the best among them, and on average it performs much better. The generated clusterings can also be used to select more informative constraints first, which further improves performance.

**Acknowledgements** We thank the anonymous reviewers for their insightful comments, which helped to improve the quality of the paper. Toon Van Craenendonck is supported by the Agency for Innovation by Science and Technology in Flanders (IWT).

## References

Adam, Antoine and Hendrik Blockeel (2015). "Dealing with overlapping clustering: A constraint-based approach to algorithm selection". In: *MetaSel workshop at ECMLPKDD*. CEUR Workshop proceedings, pp. 43–54.

Arbelaitz, Olatz, Ibai Gurrutxaga, Javier Muguerza, Jesús M. Pérez, and Iñigo Perona (2013). "An extensive comparative study of cluster validity indices". In: *Pattern Recognition* 46.1, pp. 243–256. ISSN: 00313203.

Ashtiani, Hassan, Shrinu Kushagra, and Shai Ben-David (2016). "Clustering with Same-Cluster Queries". In: *Advances in Neural Information Processing Systems 29*, pp. 3216–3224.

Bar-Hillel, Aharon, Tomer Hertz, Noam Shental, and Daphna Weinshall (2003). "Learning Distance Functions using Equivalence Relations". In: *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 11–18.

Basu, Sugato and Raymond J. Mooney (2004). "Active Semi-Supervision for Pairwise Contrained Clustering". In: *Proceedings of the SIAM International Conference on Data Mining*, pp. 333–344. ISBN: 0-89871-568-7. DOI: 10.1137/1.9781611972740.31.

Basu, Sugato, Misha Bilenko, and Raymond J. Mooney (2004). "A Probabilistic Framework for Semi-Supervised Clustering". In: *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 59–68.

Ben-David, Shai, Ulrike von Luxburg, and Dávid Pál (2006). "A Sober Look at Clustering Stability". In: *Proceedings of the 19th Annual Conference on Learning Theory*, pp. 5–19. ISBN: 3-540-35294-5, 978-3-540-35294-5. DOI: 10.1007/11776420_4.

Ben-Hur, Asa, André Elisseeff, and Isabelle Guyon (2002). "A Stability Based Method for Discovering Structure in Clustered Data". In: *Pacific Symposium on Biocomputing*, pp. 6–17.

Bilenko, Mikhail, Sugato Basu, and Raymond J. Mooney (2004). "Integrating Constraints and Metric Learning in Semi-Supervised Clustering". In: *Proceedings of the 21st International Conference on Machine Learning*, pp. 81–88.

Brazdil, Pavel B., Carlos Soares, and JoaquimPinto da Costa (2003). "Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results". In: *Machine Learning* 50.3, pp. 251–277. ISSN: 0885-6125. DOI: 10.1023/A:1021713901879.

Campello, Ricardo J. G. B., Davoud Moulavi, Arthur Zimek, and Jörg Sander (2013). "A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies". In: *Data Mining and Knowledge Discovery* 27.3, pp. 344–371. ISSN: 1384-5810. DOI: 10.1007/s10618-013-0311-4.

Caruana, Rich, Mohamed Elhawary, and Nam Nguyen (2006). "Meta clustering". In: *Proceedings of the International Conference on Data Mining*, pp. 107–118. ISBN: 0-7695-2701-9.

Davis, Jason V., Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon (2007). "Information-theoretic Metric Learning". In: *Proceedings of the 24th International Conference on Machine Learning*, pp. 209–216. ISBN: 978-1-59593-793-3. DOI: 10.1145/1273496.1273523.

Ester, Martin, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226–231.

Estivill-Castro, Vladimir (2002). "Why so many clustering algorithms: a position paper". In: *ACM SIGKDD Explorations Newsletter* 4, pp. 65–75.

Färber, Ines et al. (2010). "On Using Class-Labels in Evaluation of Clusterings". In: *Proc. 1st International Workshop on Discovering, Summarizing and Using Multiple Clusterings (MultiClust 2010) in conjunction with 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2010).*

Ferrari, Daniel Gomes and Leandro Nunes de Castro (2015). "Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods". In: *Information Sciences* 301, pp. 181 –194. ISSN: 0020-0255.

Hubert, Lawrence and Phipps Arabie (1985). "Comparing partitions". In: *Journal of Classification*, pp. 193–218. ISSN: 1432-1343. DOI: 10.1007/BF01908075.

Hutter, Frank, Holger H. Hoos, and Kevin Leyton-Brown (2011). "Sequential Model-Based Optimization for General Algorithm Configuration". In: *5th International Conference on Learning and Intelligent Optimization*, pp. 507–523. ISBN: 978-3-642-25566-3. DOI: 10.1007/978-3-642-25566-3_40.

Jain, Anil K. (2010). "Data clustering : 50 years beyond K-means". In: *Pattern Recognition Letters* 31, pp. 651–666. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2009.09.011.

Lange, Tilman, Volker Roth, Mikio L. Braun, and Joachim M. Buhmann (2004). "Stability-based Validation of Clustering Solutions". In: *Neural Computation* 16.6, pp. 1299–1323. ISSN: 0899-7667. DOI: 10.1162/089976604773717621.

Lelis, Levi and Jörg Sander (2009). "Semi-supervised Density-Based Clustering". In: *IEEE International Conference on Data Mining*, pp. 842–847. DOI: 10.1109/ICDM.2009.143.

Luxburg, Ulrike von (2007). "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4, pp. 395–416. ISSN: 0960-3174. DOI: 10.1007/s11222-007-9033-z.

Mallapragada, Pavan K., Rong Jin, and Anil K. Jain (2008). "Active query selection for semi-supervised clustering". In: *Proceedings of the 19th International Conference on Pattern Recognition*. DOI: 10.1109/ICPR.2008.4761792.

Moulavi, Davoud, Pablo A. Jaskowiak, Ricardo J.G.B. Campello, Arthur Zimek, and Jörg Sander (2014). "Density-based clustering validation". In: *Proceedings of the 14th SIAM International Conference on Data Mining*.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pourrajabi, Mojgan, Arthur Zimek, Davoud Moulavi, Ricardo J G B Campello, and Randy Goebel (2014). "Model Selection for Semi-Supervised Clustering". In: *Proceedings of the 17th International Conference on Extending Database Technology*.

Rangapuram, Syama S. and Matthias Hein (2012). "Constrained 1-spectral clustering". In: *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*.

Rousseeuw, Peter J. (1987). "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20, pp. 53–65. ISSN: 03770427. DOI: 10.1016/0377-0427(87)90125-7.

Ruiz, Carlos et al. (2007). "C-DBSCAN: Density-Based Clustering with Constraints". In: *RSFDGr'07: Proceedings of the International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing held in JRS07* 4481, pp. 216–223. ISSN: 03029743.

Shental, Noam, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall (2004). "Computing Gaussian mixture models with EM using equivalence constraints". In: *In Advances in Neural Information Processing Systems 16*.

Souto, M.C.P. de et al. (2008). "Ranking and selecting clustering algorithms using a meta-learning approach". In: *IEEE International Joint Conference on Neural Networks*. DOI: 10.1109/IJCNN.2008.4634333.

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna (2015). "Rethinking the Inception Architecture for Computer Vision". In: *Conference on Computer Vision and Pattern Recognition*.

Thornton, Chris, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown (2013). "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms". In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ISBN: 978-1-4503-2174-7. DOI: 10.1145/2487575.2487629.

Van Craenendonck, Toon and Hendrik Blockeel (2015). "Using internal validity measures to compare clustering algorithms". In: *AutoML Workshop at ICML 2015*, pp. 1–8. URL: https://lirias.kuleuven.be/handle/123456789/504712.

Vendramin, Lucas, Ricardo J G B Campello, and Eduardo R Hruschka (2010). "Relative clustering validity criteria: A comparative overview". In: *Statistical Analysis and Data Mining* 3.4, pp. 209–235. ISSN: 1932-1872. DOI: 10.1002/sam.10080.

von Luxburg, Ulrike (2010). "Clustering Stability: An Overview". In: *Foundations and Trends Machine Learning* 2.3, pp. 235–274. ISSN: 1935-8237. DOI: 10.1561/2200000008.

von Luxburg, Ulrike, Robert C. Williamson, and Isabelle Guyon (2014). "Clustering: Science or Art?" In: *Workshop on Unsupervised Learning and Transfer Learning, JMLR Workshop and Conference Proceedings 27*.

Wagstaff, Kiri, Claire Cardie, Seth Rogers, and Stefan Schroedl (2001). "Constrained K-means Clustering with Background Knowledge". In: *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 577–584.

Wang, Xiang, Buyue Qian, and Ian Davidson (2014). "On constrained spectral clustering and its applications". In: *Data Mining and Knowledge Discovery* 28.1, pp. 1–30. ISSN: 13845810. DOI: 10.1007/s10618-012-0291-9. arXiv: 1201.5338.

Xing, Eric P., Andrew Y. Ng, Michael I. Jordan, and Stuart Russell (2003). "Distance Metric Learning, With Application To Clustering With Side-Information". In: *Advances in Neural Information Processing Systems 15*, pp. 505–512.

Xiong, Sicheng, Javad Azimi, and Xiaoli Z. Fern (2014). "Active learning of constraints for semi-supervised clustering". In: *IEEE Transactions on Knowledge and Data Engineering* 26.1, pp. 43–54. ISSN: 10414347. DOI: 10.1109/TKDE.2013.22.

Zelnik-manor, Lihi and Pietro Perona (2004). "Self-tuning spectral clustering". In: *Advances in Neural Information Processing Systems 17*, pp. 1601–1608.